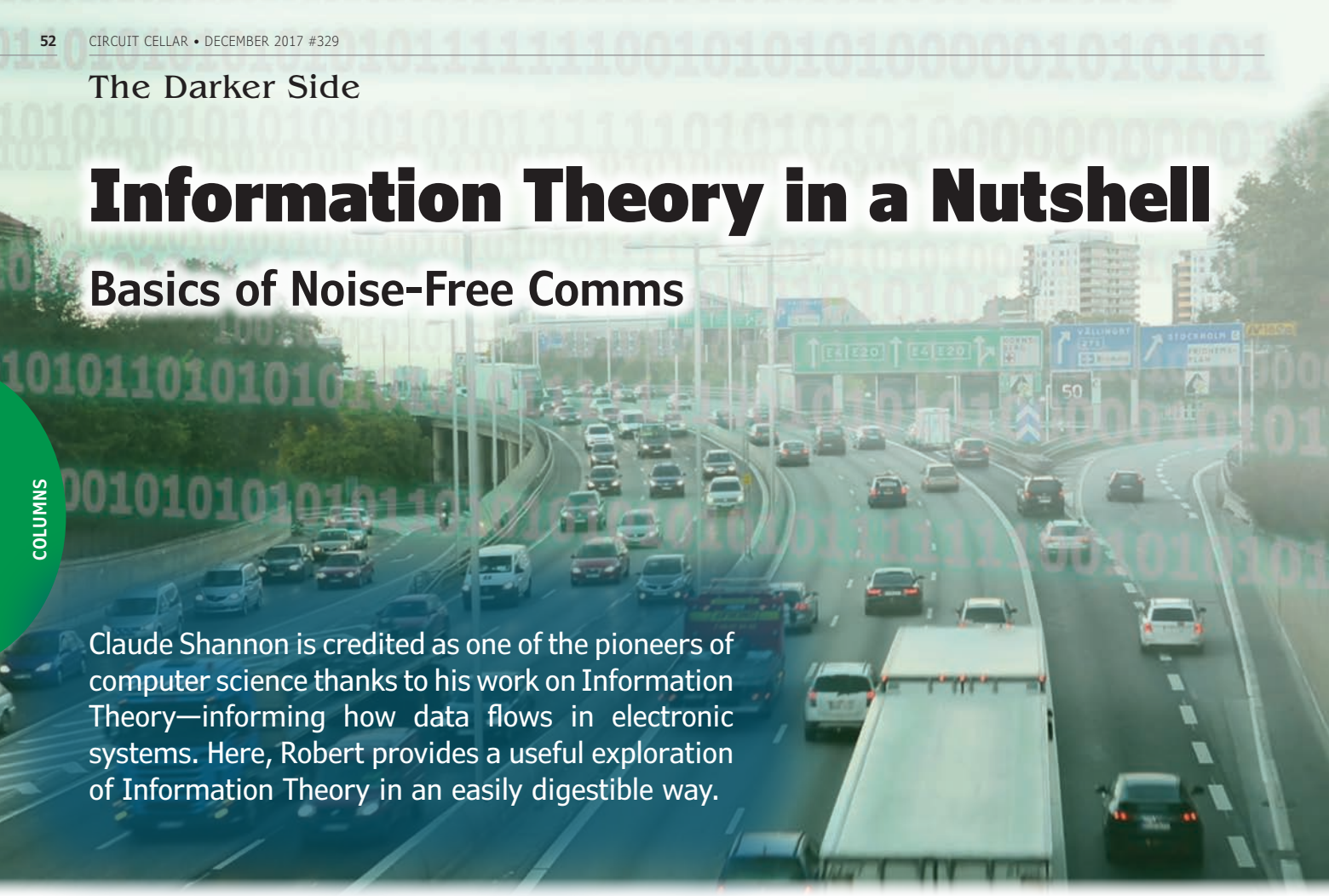
The Darker Side

Information Theory in a Nutshell

Basics of Noise-Free Comms



COLUMNS

Claude Shannon is credited as one of the pioneers of computer science thanks to his work on Information Theory—informing how data flows in electronic systems. Here, Robert provides a useful exploration of Information Theory in an easily digestible way.

By
Robert Lacoste

Anyone who's studied history knows that some long periods seem boring, while other short periods are full of amazing events. In electronics and computer science, 1948 was one of these crazy eventful periods. That year the first program was run on a computer (F. Williams, T. Kilburn and G. Toothill, University of Manchester). The transistor was invented (J. Bardeen, W. Brattain and W. Shockley, Bell Labs). And cybernetics was born, opening the way to what became artificial intelligence (Norbert Wiener). The term "bit" was invented (J. Tukey, University of Princeton). And last but not least, that same year Claude Shannon wrote his epoch-making paper on "theory of communication" that is now called information theory.

According to a paper published by the IEEE for the 50th anniversary of Shannon's work (see article materials webpage), Robert Lucky wrote: "I know of no greater work of genius in the annals of technological thought." Shannon's achievements were really ground breaking. Of course, his works were based on previous research—in particular by Hartley—and that's why his key result is now called the Shannon–Hartley theorem. Do you know any other theory that is still fundamental some 70 years later for anyone involved in communications, computer science, linguistic, biology or even cosmology (there are implications on black holes theory)? Few theoretical results had so wide a field of applications.

There is, however, a downside. Shannon was a mathematician, so his papers are full of formulas and this can be intimidating. It's useful to try to gain an understanding of what information theory is about without slogging through so much math. In this article, I'll explain it in words an embedded hardware engineer can easily grasp. This month I will cover the basics of noise-free communications, and in my next article we will see the effects of noise.

A REAL-LIFE EXAMPLE

Information theory studies the transmission and use of "information," which is quite an abstract term. I propose to focus on a simple and real-life example. Imagine that you have designed a sensor that classifies the vehicles on a highway (**Figure 1**). This sensor may be based on a camera and a Raspberry Pi module loaded with some image processing software. The software is designed to identify whether each vehicle is either a car, a truck or a bike. You have some statistics of this highway. First, you know the flow of vehicles is more or less constant—with one vehicle passing in front of the sensor each second, on average 24 hours a day and 7 days a week.

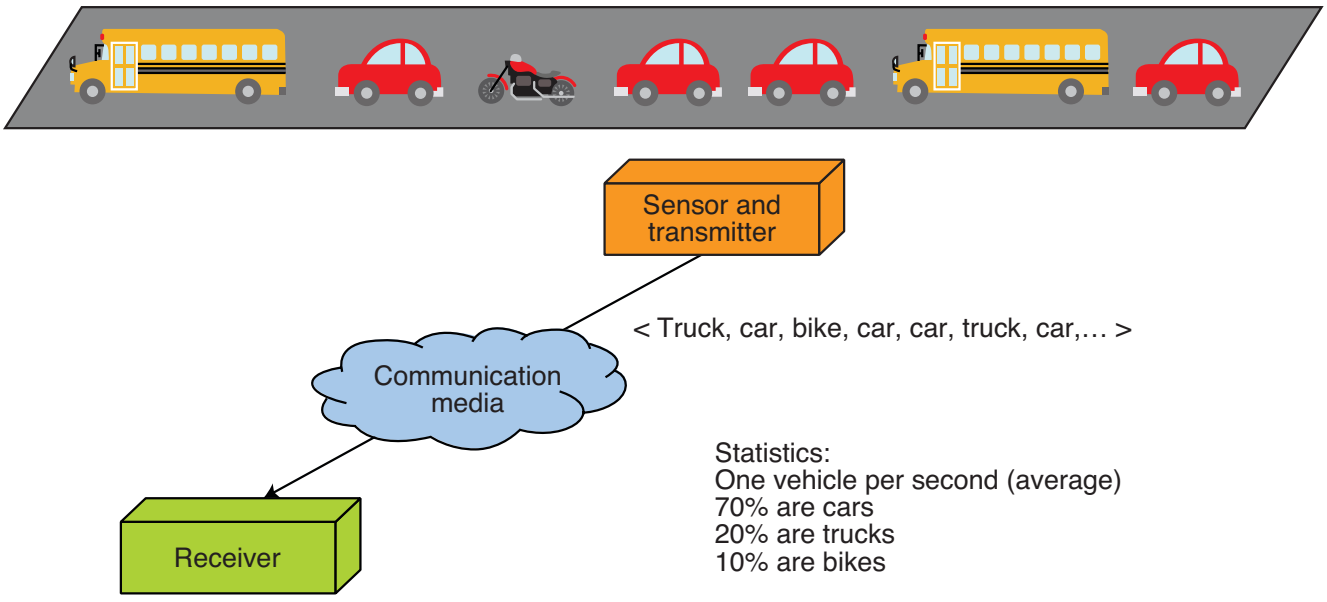


FIGURE 1
In this simple example, the source of information needs to send whether each vehicle is a car, truck or bike, with known statistics.

Okay, that is probably far more simple than real life, but this will help simply my example. Moreover, you have statistics that show that 70% of the vehicles on this highway are cars, 20% are trucks and the remaining 10% are bikes—all that still on average of course.

Imagine that your job is to build a transmission system to send the resulting flow of information to a remote server. For each vehicle passing in front of the sensor, you just need to indicate which kind of vehicle it is: car, truck or bike. This transmission link is, say, a digital line able to send 1s and 0s. However, let's assume that this remote system is very far away and that the communications are costly. As a result, you must reduce as much as possible the number of bits transmitted over the link in order to reduce transmission costs. What solutions could you use? What is the minimum information flow required, in terms of bits per second (bps)? As I will explain shortly, that's exactly what information theory does best.

BASIC CODING

For the moment, I'll assume that the transmission channel is error free. In other words, each bit transmitted is received without error. With that assumption, the required number of bits per second is of course simply linked to the encoding method used to indicate "car," "truck" or "bike". Some methods are

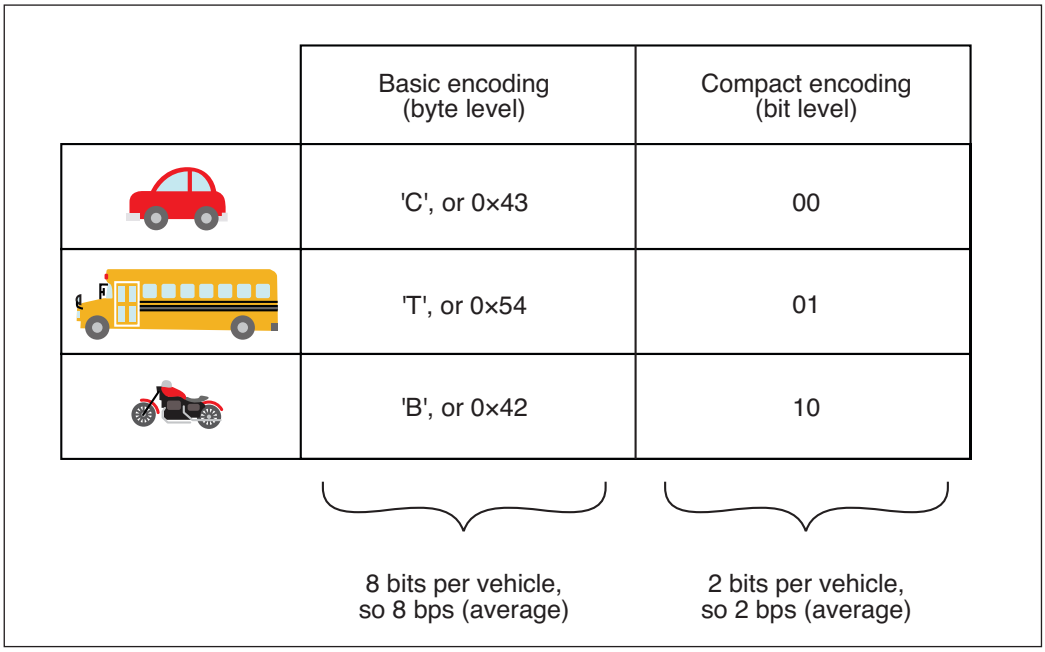


FIGURE 2
With basic encoding you can get down to 2 bits per vehicle on average.

COLUMNS

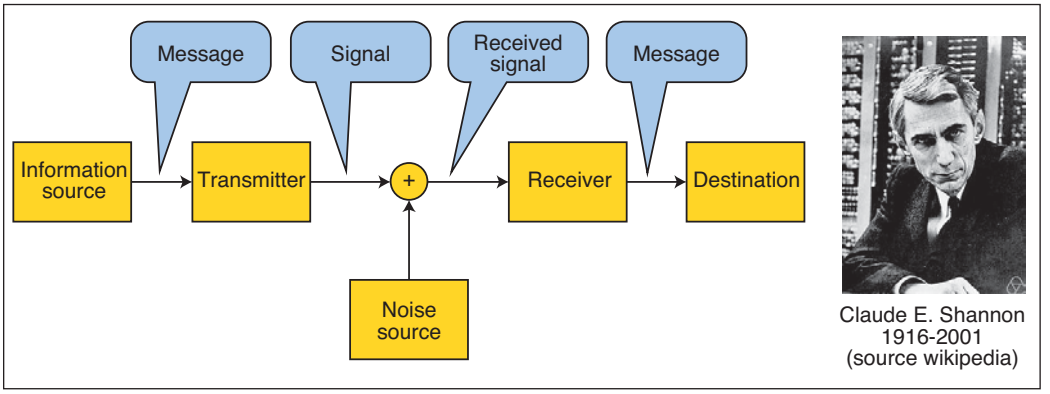


FIGURE 3
The model established by Shannon is very generic.

far more efficient than others. One of the less efficient in that context is probably HTML (sorry web aficionados!). Whenever there is a car, you would send some HTML strings like "<html><body> Car </body></html>" and similar strings of bytes for trucks and bikes respectively. That would be 34 bytes for each vehicle, or 34 x 8 = 272 bits each second—meaning a required transmission throughput of 272 bps on average. I'm sure you're smart enough to see immediately that there are better ways to proceed. One of the simplest ways is to encode each kind of vehicle using a single ASCII character. Then you just send this character as a simple byte over the communication link. For example, you could send 'C' for a car, 'T' for a truck and 'B' for a bike. One byte is 8 bits, so you have reduced the channel requirements to 8 bits for each vehicle, so 8 bps on average—not a small improvement.

Can you do better? Of course, nothing prevents you from using more compact encoding methods (**Figure 2**). Here we have three different values to encode, so we could use only two bits per vehicle: 00 for a car, 01 for a truck and 10 for a bike for example. You would have still 11 as a spare. This is a little more complex to implement as computers prefer to work on bytes than on individual bits—but it is possible. The required channel capacity reduces down to 2 bps on average.

SHANNON TO THE RESCUE

Now comes the million-dollar question: Is it possible to find clever encoding solutions and to reduce the required channel capacity to less than 2 bps? Moreover, what would be the absolute minimum channel capacity? That's the problem Claude Shannon solved in 1948—both for noise-free channels but also in case of noise.

Shannon modeled the problem using the very generic architecture illustrated in **Figure 3**. First, there is an information source that generates messages made of symbols. Each symbol may have a different occurrence probability, but the source is supposed to be random. More exactly, the source is defined as "ergodic"—meaning that it respects certain statistical properties like generating any possible sequences during a long period of time. From an engineering standpoint consider it as nearly equivalent to "random." Then there is a transmitter that translates the symbols into something that could be sent over a transmission channel. This transmitter could use compression, modulation or any other method. Some noise could be added to the channel, and then a receiver tries to recover the resulting symbols and to forward them to the destination node.

Using this model, Shannon defined a value called the "entropy of the source." This entropy is actually a measure of the information content of each symbol sent by the source. It is based on the statistical properties of the different symbols, and is expressed in bits. Don't confuse this entropy with the entropy used in thermodynamics—even if there are links. How is it calculated? It's done very easily as follows:

Entropy of an ergodic source:

$$E = -\sum p_i \log_2(p_i) = -\sum p_i \frac{\log(p_i)}{\log(2)}$$

ABOUT THE AUTHOR

Robert Lacoste lives in France, between Paris and Versailles. He has 30 years of experience in RF systems, analog designs, and high speed electronics. Robert has won prizes in more than 15 international design contests. In 2003 he started a consulting company, ALCIOM, to share his passion for innovative mixed-signal designs. Robert's bimonthly Darker Side column has been published in Circuit cellar since 2007. You can reach him at rlacoste@alciom.com.

How do you use this formula?

First, you need to know how many different symbols could be generated by the source. In my example, there are three: car, truck or bike. Then you need to know what is the respective probability of each symbol. Remember that I considered that 70% of the vehicles are car, so the probability of the "car" symbol is p_{car}=70/100=0.7. Similarly, the probability of "truck" and "bike" are 0.2 and 0.1 respectively. The sum 0.7+0.2+0.1 equals 1. Then you can calculate the entropy of the source with the following formula:

$$E = -p_{car} \times \log_2(p_{car}) - p_{truck} \times \log_2(p_{truck}) - p_{bike} \times \log_2(p_{bike})$$

In this formula, the only unusual thing is the log2 function. This is a logarithm calculated using base 2. You will very probably not have this function on your calculator, but you can easily use a standard log function as follows:

$$\log_2(x) = \frac{\log(x)}{\log(2)}$$

In our example:

$$E = -\left(0.7\frac{\log(0.7)}{\log(2)} + 0.2\frac{\log(0.2)}{\log(2)} + 0.1\frac{\log(0.1)}{\log(2)}\right)$$

So:

$$E = -(-0.360 - 0.464 - 0.332) = 1.157$$

You will find an entropy of E=1.157 with the values of our example. Now the magic can happen. Shannon proved that this entropy is exactly what you are looking for! It is equal to the minimum required capacity of any transmission channel able to send these symbols. Once again, this assumed that there are no transmission errors (noiseless channel). This means that in our example there is a way to transmit the flow of information with just 1.157 bits per symbol. So, in average the absolute minimum is 1.157 bps as we have one symbol per second to send. This also states that there is no way to use less—even if you are very clever.

This result is true for any kind of symbols. It could be a choice of 1 over 3 like in my example, English words, audio samples or anything else. Maybe another small calculation would help you to understand. What would be the entropy of a binary source, generating 0s and 1s

	Probability	Variable length coding
	0, 7	0
	0, 2	10
	0, 1	11

1 or 2 bits per vehicle,
In average: 1 bit × 0, 7 + 2 bits × (0, 2 + 0, 1) = 1.3 bit
so 1.3 pps (average)

FIGURE 4
This shows a basic variable length encoding that reduces the channel requirement down to 1.3 bits per vehicle on average.

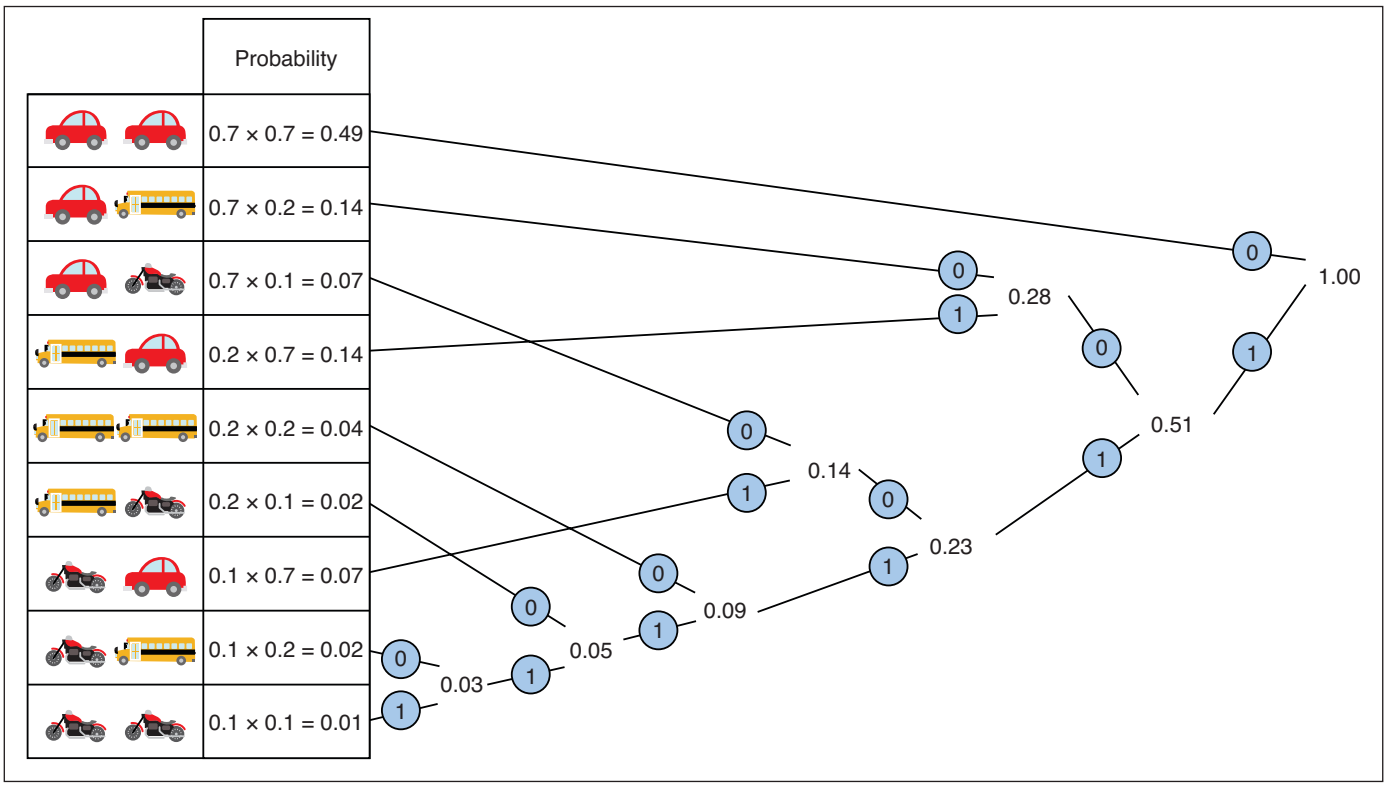


FIGURE 5
Here is the second-order Hamming encoding algorithm. It starts with all possible groups of two symbols, grouped two by two starting with the lowest probabilities.

with 0.5 and 0.5 probability? Here's the math: - 0.5 x $\log_2(0.5)$ - 0.5 x $\log_2(0.5)$. Solve that and you will find that the resulting entropy is 1 bit per message, as you may expect. If you have a binary source but with one symbol occurring far more often that the other, then the entropy will be lower. Do the calculation to convince yourself. This means that the information content is lower.

VARIABLE LENGTH CODING

Shannon's theorem is unfortunately not a so called constructive proof. It states that such a coding method must exist, but doesn't say which one. In case of a noiseless channel, one of the most efficient encoding method is called variable length encoding. The idea is very simple. With the 2-bit encoding solution described previously, I used 2 bits for each symbol, even though the "car" symbol is far more frequent than the other ones. The encoded bit flow would be lighter if the most frequent symbols were encoded with less bits than the rarest ones. By the way, this is exactly what Samuel Morse did when he encoded the most frequent letters with short sequences of dots and dashes. For instance, E is only one dot in Morse code, and it is the most frequent letter in English.

How do you implement such a variable length coding method in our example? The simplest solution is illustrated in **Figure 4**. Here the "car" symbol is the most frequent, so let's encode it using a single '0' bit. The two other symbols are then encoded with a 2-bit sequence starting with a '1'. So, '10' for trucks and '11' for bikes. That way only one bit is required for 70% of the symbols, whereas 2 bits are needed for the remaining 30%. On average, there is then 0.7 x 1 + 0.3 x 2 = 1.3 bits per symbol. This means that on average only 1.3 bps will be sent through the link. Knowing that encoding rule, you can then easily recover each symbol on the receive side—as long as there are no errors.

It is now time to see the big picture. We have an example of an information source with an entropy of 1.157 bits. Using simple encoding methods, we get down to 2 bps. With a variable length coding, we've reduced the required bit flow down to 1.3 bits per symbol on average.

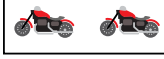
IT'S HAMMING TIME

Of course, 1.3 is higher than 1.157. And Shannon told us that the entropy is the absolute minimum. You are probably wondering how to find a more efficient method, meaning one that will get as close as possible to 1.157 bits per symbol? In the case of a noiseless channel, the solution is called Hamming encoding. In fact, the Hamming algorithm allows you to construct variable-length codes as close as possible to Shannon's limit as you wish, but only with error-free channels.

To be honest, the variable-length encoding method I showed you previously was in fact the simplest Hamming encoding solution,

For detailed article references and additional resources go to: www.circuitcellar.com/article-materials

based on single symbols. Let's call that Order-1 Hamming. To find a more efficient solution, you need to consider groups of symbols together. I will show you the slightly more complex but more efficient Order-2 Hamming encoding. It is illustrated in **Figure 5**. Here you need to consider all possible groups of two successive symbols—car-car, car-truck and so on. If you think about it, there are 3 x 3 = 9 such groups. Because

Bi-symbol	Probability	Huffmann code	Bit length	Bit length × probability
	0.7 × 0.7 = 0.49	0	1	1 × 0.49 = 0.49
	0.7 × 0.2 = 0.14	100	3	3 × 0.14 = 0.42
	0.7 × 0.1 = 0.07	1100	4	4 × 0.07 = 0.28
	0.2 × 0.7 = 0.14	101	3	3 × 0.14 = 0.42
	0.2 × 0.2 = 0.04	1110	4	4 × 0.04 = 0.16
	0.2 × 0.1 = 0.02	11110	5	5 × 0.02 = 0.10
	0.1 × 0.7 = 0.07	1101	4	4 × 0.07 = 0.28
	0.1 × 0.2 = 0.02	111110	6	6 × 0.02 = 0.12
	0.1 × 0.1 = 0.01	111111	6	6 × 0.01 = 0.06

Average = 2.33 Bits per symbol
so 2.33/2 = 1.165 bits per vehicle
so 1.165 in average

FIGURE 6
Reading the binary tree illustrated in Figure 5 from right to left gives the encoding of each couple of symbols. The average channel capacity is now down to 1.165 bps—very close to the theoretical limit stated by Shannon.

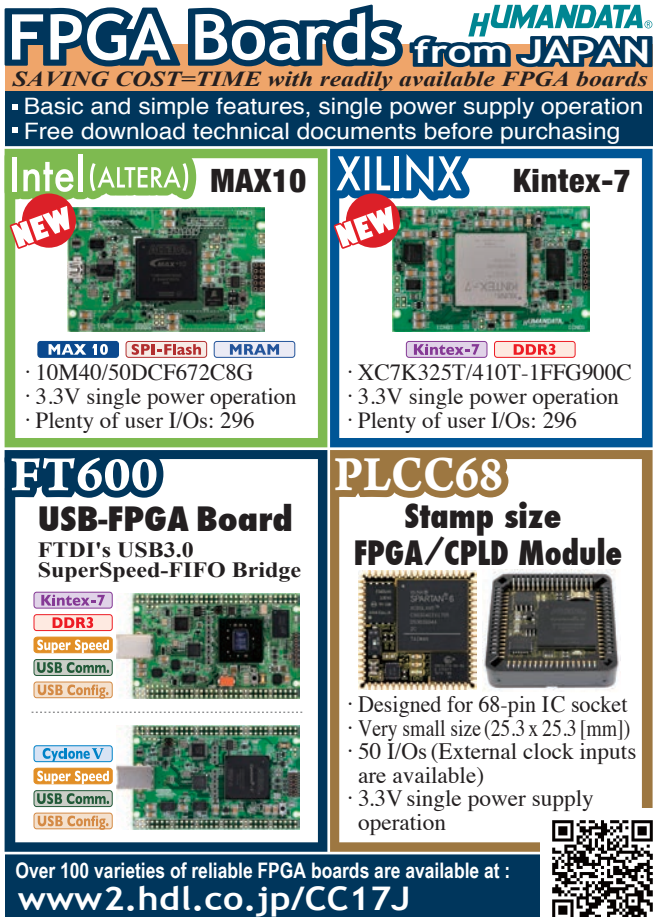
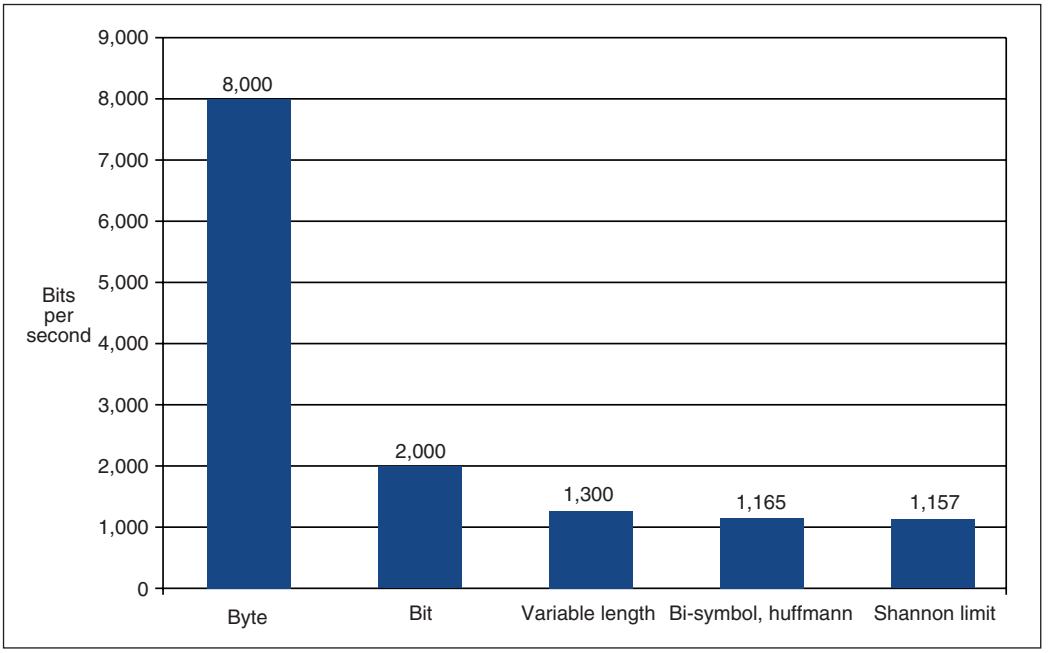


FIGURE 7
A short summary of the improvements covered in the article



the source is supposed to be ergodic, the probability of seeing each couple of symbols is the product of both probabilities. For example, the probability of occurrence of the "car-truck" sequence is the probability of "car" times the probably of "truck", or 0.7 x 0.2 = 0.14. These two-symbol probabilities are shown on Figure 5.

How to find the best variable length encoding method? First, find the two configurations with the lower probability. Here they are respectively "bike-truck" (0.02) and "bike-bike" (0.01). Group them together, and label the first one with a binary '0' and '1' for the other. Then calculate the probability to see either of them. This is the sum of their probabilities: 0.02 + 0.01 = 0.03.

As illustrated in Figure 5, you simply continue the same work from left to right for all symbols or groups of symbols. At each step, select the two with the lowest probabilities, group them, label both branches with 0 and 1, and calculate the sum probability. If you do this until you have included all symbols, you end up with a binary tree. Magic again! This tree provides an optimal way to encode each group of two symbols with the lowest possible number of bits. You just have to read the binary tree from right to left! Look again on Figure 5. Consider for example the "car-car" sequence—the most frequent combination—is encoded with a single '0' bit, whereas the "car-bike" sequence is encoded as '1100'. In contrast, the "bike-bike" sequence—a very infrequent combination—is encoded as '111111' requiring 6 bits.

What is the resulting bit rate on the transmission channel if we use this encoding method? It's very easy to calculate. This the sum of the number of bits of each configuration times their respective probabilities. The calculation is shown in **Figure 6**. On average, this method requires 2.33 bits as shown. But wait! This is higher than the 1.3 bit we got with the previous method, right? Yes. But remember we're encoding two symbols at a time. As a result, it requires 2.33/2 = 1.165 bits per input symbol. Are you impressed? You should be given that we're are now very close to the minimum channel capacity calculated using Shannon's formula (1.157) (**Figure 7)**. Of course, 1.165 is higher than the 1.157 that Shannon gives an absolute minimum—but we are close. Doing the same work with longer sequences of 3 or 4 symbols will bring you closer to this limit. You can theoretically get as close as you want but with more and more complexity. Try it as an exercise using groups of 3 symbols. There are just 27 of them!

WRAPPING UP

I have tried to show the basics of the Shannon–Hartley theorem in the case of error-free transmissions. The theorem is even stronger and more impressive in case of noise, but that will be the subject of my next article. Take a look at Shannon's original paper. It is far easier to read than other mathematical works. If you want a simpler—but excellent and detailed—presentation of the information theory and its applications, buy the excellent book written by John R. Pierce. Its 1st edition was written in 1961 and the 2nd in 1980 but it is still a great reading (see article materials webpage). Have fun!

The Darker Side

Shannon and Noise

Putting the Theorem to Work

COLUMNS

In his previous article on Information Theory, Robert explained the Shannon–Hartley theorem as applied to the case of error-free transmissions. This time he applies the theorem to noise, for which the theorem is even stronger and more impressive.

By
Robert Lacoste

Welcome back to The Darker Side. My previous article was about the great results established by Claude Shannon in 1948. In that article—"Information Theory in a Nutshell" (*Circuit Cellar* 329, December 2017)—I dealt with noise-free transmissions and I promised that this month I'd talk about the more interesting noisy channels. Well, here we go! Take a seat and enjoy. As always, I will use as little math as possible so you can grab the fundamentals.

In case you've either missed my last article or you need a refresher, here's a summary: Information theory studies the transmission of data between devices, through a given transmission channel. Information theory is based on math, so it provides very strong and generic results. Thanks to its wide scope, it has applications ranging from computer science to biology to cosmology and—fortunately—electronics.

In my last article, I assumed that the transmission channel was error free: Every bit transmitted was supposed perfectly received. In that case, the so-called Shannon's "noiseless coding theorem" (also called "source coding theorem") allows you to calculate the minimum channel capacity in bits per seconds (bps), depending on the source. More exactly, Shannon assumes that the source generates on average S symbols per second. Each symbol is taken from a set of N different symbols. Symbols could be groups of bits, English words or anything else. And finally, each symbol has a given probability p_i to appear—meaning that some symbols could be sent more often than others. Assuming that the source has some good statistical properties (experts talk about "ergodic source"), Shannon defined a value called the entropy E of the source, calculated as:

$$E = -p_1 \times \log_2(p_1) - p_2 \times \log_2(p_2) - p_3 \times \log_2(p_3) - \text{etc.}$$

In that formula, $\log_2$ is the logarithm function calculated using base 2, easily calculated as:

$$\log_2(x) = \frac{\log(x)}{\log(2)}$$

Ok, nothing exciting there…until you know the first strong theorem established by Shannon. It states that this entropy E is in fact closely linked to the minimum required channel capacity to transmit the data flow:

$$C_{MINIMUM}(\text{bits/s}) = E(\text{entropy per symbol, in bits}) \times S(\text{symbols/s})$$

Here's an example: If the source sends S = 1,000 symbols per second and if the calculated entropy of the source is E = 1.5 bits per symbol, then the minimum capacity of the channel is 1,500 bits per second. This means that there is a way to correctly encode the data flow such that 1,500 bps are enough, on average. Read again in my last article where I showed that the so-called Hamming encoding enables you to come as close as you want to

this limit. Conversely this theorem also states that there isn't any way to reduce it to less than this limit—even if you are the cleverest engineer on Earth!

In this article I will no longer assume that the channel is noise free. For simplicity, I will however restrict myself to the case of a binary data source (there are only two symbols to transmit: 1 and 0). I will also assume that the probability of 1s and 0s is identical: 50% of the bits are 1s and 50% are 0s. It you do the calculation of the entropy of such a source, using the formula above, you will find of course that E = 1. This means that such a source needs a channel with a capacity of 1 bit per symbol: C = 1 x S = S. This simply means that there isn't any way to compress a random source of binary bits.

SOME NOISE?

Okay, now let's see what happen when there is some noise. And there is always noise in a circuit or transmission. Trust me! First of all, noise introduces errors in the bit transmission. Secondly, noise will limit the capacity of the channel, depending on its bandwidth. That second sentence may seem strange, but don't worry. I will come back to it.

For the moment, I will simply assume that the noise introduces some errors on the transmitted bits. Of course, the higher the noise, the higher the probability to receive a wrong bit—as illustrated on **Figure 1**. By the way, I devoted an article on bit errors a while

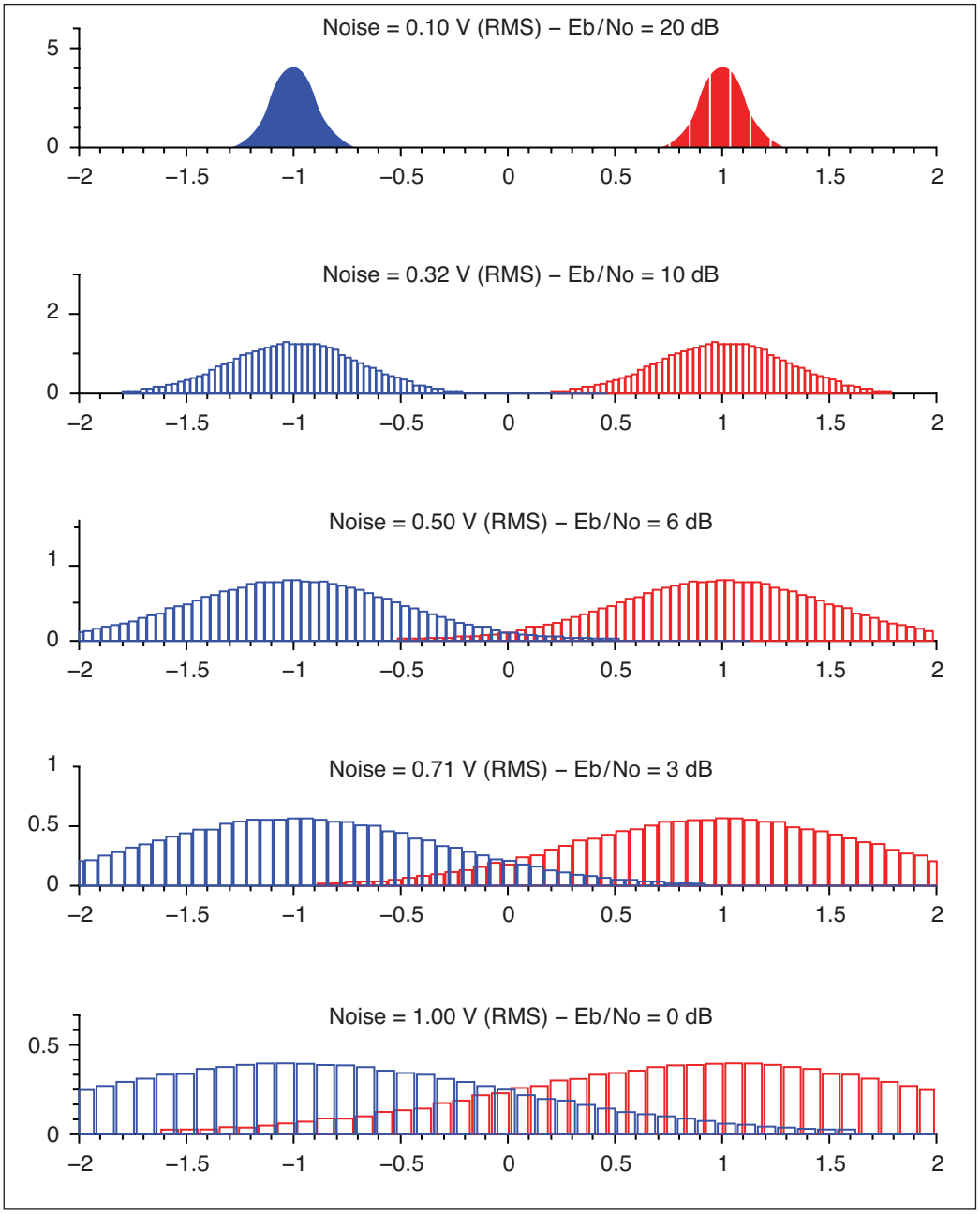

Noise = 0.10 V (RMS) – Eb/No = 20 dB
Noise = 0.32 V (RMS) – Eb/No = 10 dB
Noise = 0.50 V (RMS) – Eb/No = 6 dB
Noise = 0.71 V (RMS) – Eb/No = 3 dB
Noise = 1.00 V (RMS) – Eb/No = 0 dB
–2
–1.5
–1
–0.5
0
0.5
1
1.5
2

FIGURE 1
When noise increases, the probability of an error on a single bit also increases. Here the noise is supposed gaussian, which implies that there are always some errors from time to time.

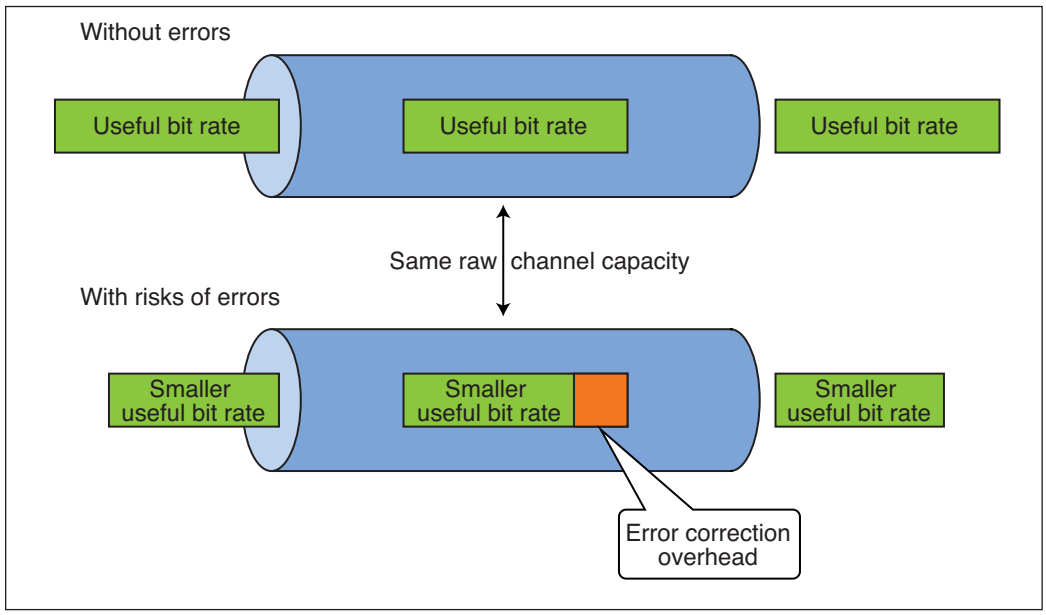


FIGURE 2
Error correction protocols always reduce the available throughput, or increase the required raw channel capacity for the same useful throughput.

ago called "Let's Count Errors" (*Circuit Cellar* 295, February 2015).

If we assume that the probability of a bit error p_{err} is not null, then the transmission will be deteriorated. Now, assume that you need to design a reliable and error free transmission using such a channel. What would your options be? You could try to improve the electronics with a better signal to noise ratio (SNR), reducing the noise as much as you can or try using stronger signals. Another way—more interesting from an information science point of view—would be to use an error correction protocol instead of just sending the raw bits. Such a protocol would help to recover the erroneous bits—as long as they're not too numerous. For that, you could simply add checksums to the transmitted data and retransmit any wrongly received message. You could also use forward error correction codes, as explained in another of my old articles: " Living with Errors: An Introduction to Forward Error Correction," (*Circuit Cellar* 235, February 2010).

ABOUT THE AUTHOR

Robert Lacoste lives in France, between Paris and Versailles. He has 30 years of experience in RF systems, analog designs, and high speed electronics. Robert has won prizes in more than 15 international design contests. In 2003 he started a consulting company, ALCIOM, to share his passion for innovative mixed-signal designs. Robert's bimonthly Darker Side column has been published in Circuit cellar since 2007. You can reach him at rlacoste@alciom.com.

In either case, the addition of these error correcting protocols will inevitably increase the number of bits to be transmitted. You will have to transmit not only the source bits, but also additional bits like checksums, retransmissions, error correcting codes and so on (**Figure 2**). So, when noise introduces some errors in the transmission, the actual channel capacity must be higher to allow error recovery. More exactly, this capacity (in bits per seconds) must be more than the minimum capacity in the case of a noise free channel, which is the entropy of the source times the symbol rate. But how much more?

FUNDAMENTAL THEOREM

And here comes some magic again with the fundamental Shannon theorem in presence of noise. Shannon proved that the minimum channel capacity in the presence of such a noise must be:

$$C_{MINIMUM}(\text{bits/s}) = E(\text{entropy per symbol, in bits}) \times \frac{S(\text{symbols/s})}{R(\text{reduction factor})}$$

or

$$C_{MINIMUM}(\text{bits/s}) = \frac{B(\text{useful bit rate, bits/s})}{R(\text{reduction factor})}$$

In this formula, R is called the reduction factor, and can be easily calculated with the following function:

$$R = 1 + p_{err} \times \log_2(p_{err}) + (1 - p_{err}) \log_2(1 - p_{err})$$

As you can see, this number is only dependent on the bit error rate. It is always

between 0 and 1. Look at the plot of this function on **Figure 3**. The capacity of a channel in presence of noise is simply reduced by a factor R, depending on the error probability. An example should help: Assume that you send ten million binary symbols (bits) per second and that 5% of them are corrupted by noise. What would be the minimum channel capacity for an error free transmission? Here we have E = 1 (one bit per symbol), S = 10M and R = 1+0.05.$\log_2$(0.05) + 0.95.$\log_2$(0.95). Do the calculation and you will find R = 0.71. That means that the minimum channel capacity in that case is:

$$C_{MINIMUM} = 1 \times \frac{10M}{0.71} = 14.1\,Mbps$$

Okay, but what does this actually mean? What did Shannon prove? First, he demonstrated that if the raw channel capacity is lower than this value then there is absolutely no way to invent an error correction protocol that will correct the transmission errors. You will always have a significant number of errors, even if you are a genius. Conversely, Shannon also demonstrated that there exists a solution to have an error rate as low as you want, as long as the channel capacity is just slightly above that threshold. That result astonished the community in 1948—and is still to this day nearly unbelievable. The bad news is that the Shannon theorem is not a so called constructive proof. It doesn't tell you what this best error correction method is, but simply that it exists. With that in mind, thousands of researchers are working on error correcting codes since 1948, in order to find error correction codes as close as possible to Shannon limit!

To summarize, the quite simple formula given above enables you to calculate the Shannon limit or Shannon capacity of a communications channel, which is the theoretical maximum error-free information transfer rate of the channel for a particular noise level. Information simply cannot be guaranteed to be transmitted reliably at rates beyond this channel capacity.

LIMITED BANDWIDTH

In my introduction, I told you that noise also limits the capacity of the channel depending on its bandwidth. What's that all about? To understand, we have to move from a theoretical transmission channel to something closer to an actual channel—like a wire or radio link. It will for sure have a limited bandwidth W, expressed in Hertz. Let's assume W = 100 kHz, and for simplicity's sake consider that the channel is something like a brick-wall filter: It transmits perfectly all frequencies from DC to 100 kHz, but stops any signals above that threshold.

What's the capacity of such a channel in bits/s? First, a strange but true result: If there isn't any noise, then this capacity is infinite. Now you probably think I'm crazy. Hey

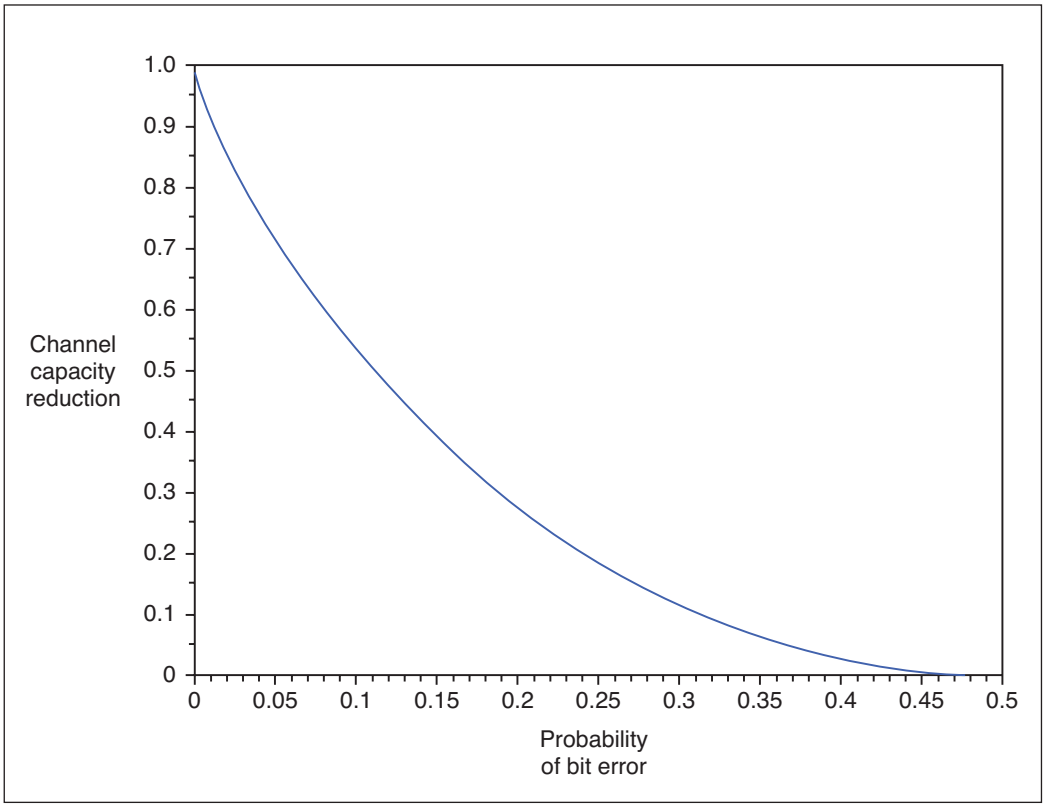
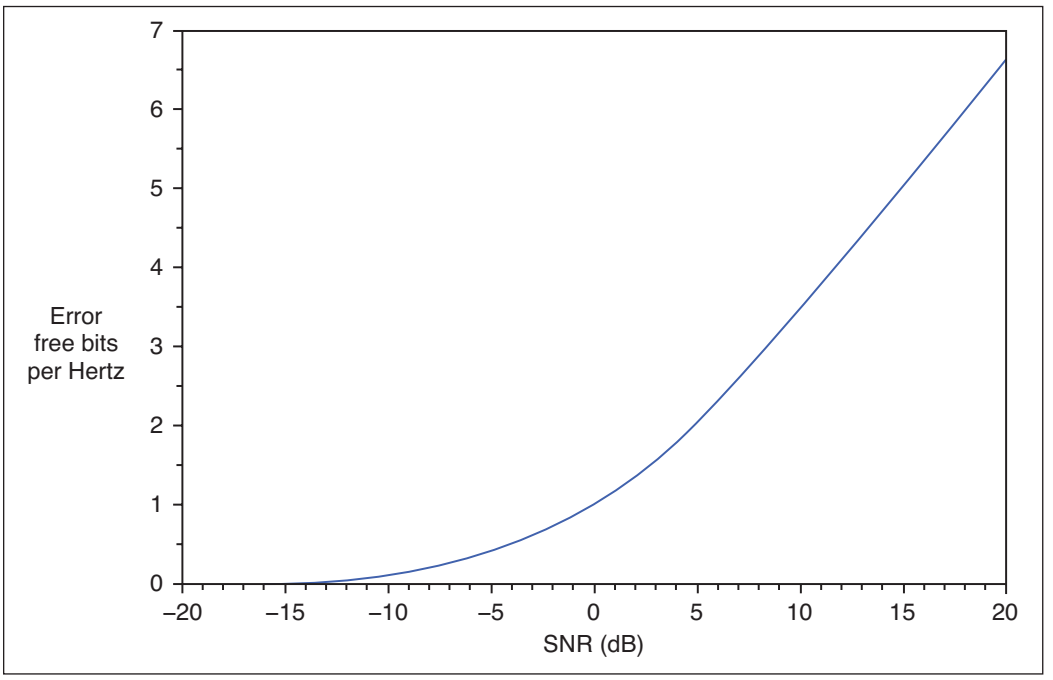


FIGURE 3
This plot shows you the reduction of the channel capacity R, depending on the probability of an error on single bits: p_{err}. Not surprisingly, the throughput is 100% when there are no errors, and 0 when bits are random (50% of bits wrong).

FIGURE 4
Here is the maximum useful error-free bit rate B_{max} for 1 Hz of bandwidth, depending on the signal to noise ratio SNR (here expressed in dB).



Robert, don't you remember Nyquist limit? As the channel bandwidth is W then one can send only send 2xW independent samples per second on the wire. It is the same rule applies when you state that a 200 ksample/s analog-to-digital converter (ADC) can only manage signals up to 100 kHz. That is true, but if there is absolutely no noise then each sample can have theoretically an infinite resolution! You could for example group the bits to be transmitted 100 per 100, convert each group into an analog voltage with a 100-bit ADC (remember, there isn't any noise at all so this is theoretically possible) and send that voltage 200k times per second. You would get 100 x 200k = 20 Mbps. If you need more, just increase the ADC resolution. This is exactly the idea behind what is called pulse code modulation (PCM), invented in 1937 by A.H Reeves (French Patent 852,183).

So, you understood that the actual capacity of a channel is in fact limited by noise—or more exactly by the signal to noise ratio (SNR). Now let's look at another form of the Shannon limit formula, usually called the Shannon–Hartley theorem. It could be derived from the formulas above but just take it as granted. Assuming that the noise is a "good" noise (gaussian), then the maximum number of useful error-free bits per second B_{max} is:

$$B_{max}(bits/s) = W(Hertz) \times \log_2(1+SNR)$$

Figure 4 shows you the behavior of this function but I am sure you want some examples. If the signal to noise ratio is 1 (same signal power as noise power), then:

$$B_{max} = W \times \log_2(1+1) = W$$

In that case you can transmit 1000 error-free bits per second in a 1,000 Hz channel bandwidth. However, if the signal to noise ratio is far better—say if the signal is 100 times more powerful than noise (20 dB ratio)—then:

$$B_{max} = W \times \log_2(1+100) = 6.65 \times W$$

In the same 1,000 Hz bandwidth, you could then transmit 6,650 bits per second without error!

EB/N0?

I have covered all the important aspects of Shannon's theorem, but here's some additional information for those who want to go a little deeper. Just jump over this section if you're feeling overloaded with math.

If you open any book on signal processing or information theory, you will find the expression "Eb/N0" in every page. What is this strange thing? It's quite simple in fact. But I must admit that it took me some time to understand it and to link it to the formulas above. Here is my personal summary: Eb is the energy per bit (in Joules), calculated for each useful transmitted bit. It is then equal to the signal power S (in Watts) divided by the useful bit rate B:

$$Eb = \frac{S(\text{signal level})}{B(\text{useful bit rate, error free})}$$

Similarly, N0 is called the noise power spectral density ratio. It is the noise power

For detailed article references and additional resources go to: www.circuitcellar.com/article-materials

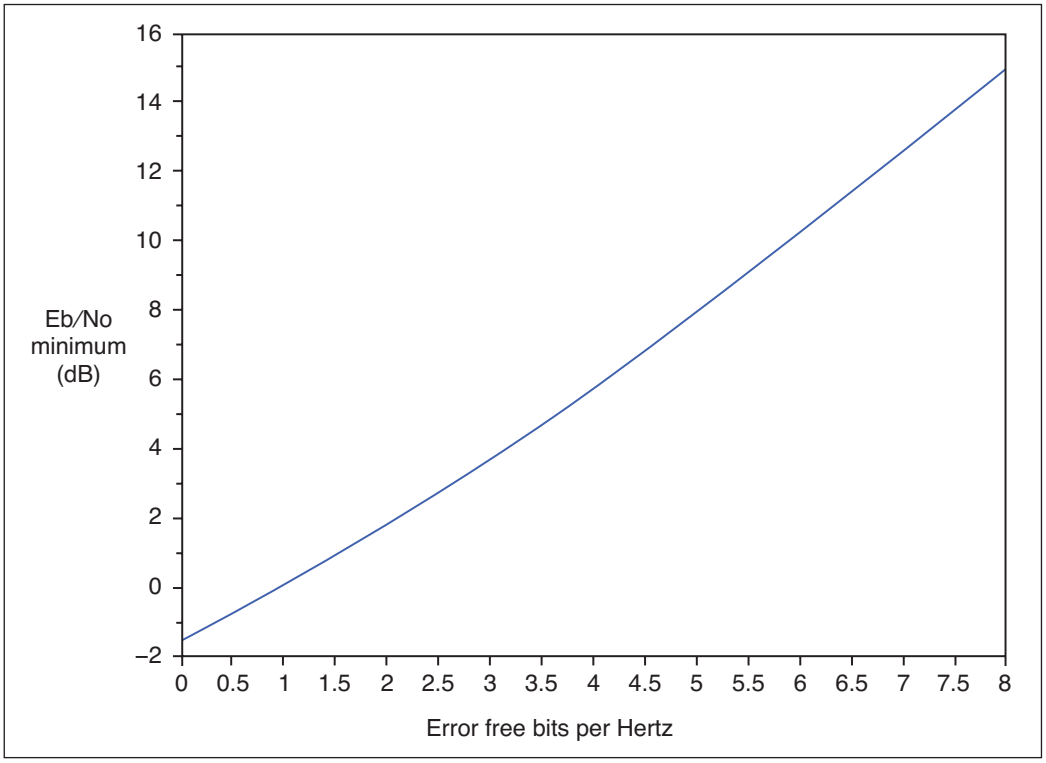


FIGURE 5
The maximum achievable Eb/N0 performance is closely linked to H: the number of useful error-free bits per Hertz of bandwidth.

in a 1 Hz bandwidth, measured in Watts per Hertz. N0 is then calculated as the total noise power N divided by the channel bandwidth W:

$$N0 = \frac{N(\text{noise level})}{W(\text{channel bandwidth})}$$

You will notice that Eb and N0 are both expressed in Joules, so their ratio is dimensionless. It can then be expressed in decibels if you wish. Of course, Eb/N0 is closely related to the carrier-to-noise ratio:

$$SNR = S/N = \frac{(Eb \times B)}{(N0 \times W)} = (Eb/N0) \times (B/W)$$

And finally, remember the Shannon capacity limit? It is calculated as:

$$B_{max}(\text{bits/s}) = W(\text{Hertz}) \times \log_2(1 + SNR)$$

The useful bit rate B should stay below the threshold B_{max}, so we must have:

$$B < W \times \log_2(1 + SNR) \text{ or}$$
$$B/W < \log_2(1 + Eb/N0 \times B/W)$$

If we call H = B/W (useful bits per Hertz of bandwidth), then this leads to:

$$H < \log_2(1 + Eb/N0 \times H) \text{ or}$$
$$2^H < 1 + Eb/N0 \times H$$

or finally:

$$Eb/N0 > \frac{(2^H - 1)}{H}$$

Read this last formula again. It provides an absolute minimum on Eb/N0, depending only on the effective number of bits per Hertz of bandwidth. This is illustrated on **Figure 5**. If H is very small (large bandwidth), then Eb/N0 gets close to ln(2), which in decibels is 10log(ln(2)) = -1.59 dB. This often-quoted limit of Eb/N0 = –1.59 dB applies only to the very theoretical case of infinite bandwidth. For limited bandwidth, the absolute Eb/N0 limit is always higher. For example, if H = 2 (bandwidth twice more than the bit rate), then the formulas shows that Eb/N0 is always above 1.76 dB.

WRAPPING UP

I know that these two articles on information theory were a little more math-oriented than usual, but I am strongly convinced that these results are far too important to be skipped by any electronic engineer, ham radio guy or telecommunication enthusiast. I hope that you have grabbed the essentials. If not, read these articles again and have a look on the books and articles listed on *Circuit Cellar's* article materials webpage. And finally, like most of you, I'm far more of an electronic designer than an information science expert, so don't hesitate to contact me if I've said something wrong or too simplified. Last but not least, play with these formulas yourself and have fun!